TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Raul Vesinurm 190510IADB

FITLAPI KASUTAJATE TOITUDE TELLIMISE JA MÜÜMISE VEEBIRAKENDUS

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Raul Vesinurm

03.01.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua töötav veebirakendus FitCook, kus Fitlapi kasutajad saavad ühineda keskkonnaga, mis abistab kasutajaid toidu tegemisel. Rakenduse abil kohtuvad kaks osapoolt. Need, kes soovivad, et neile toit valmis tehakse vastavalt enda energiavajadusele ning need, kes soovivad süüa teha.

Arendusprotsessi käigus luuakse veebirakendus, mis annab tööd kodukokkadele, kes koostavad erineva pikkusega menüüsid Fitlapi toitudest. Inimene, kes soovib, et talle süüa tehakse liitub talle meeldiva kokaga ning valib toidud menüüst, mida soovib. Sealjuures kokk arvestab inimese kaloraažiga ning vastavalt sellele pakendab õige koguse toidu pakendisse, millele saab klient järgi tulla.

Lõputöö katab endas veebiteenuse ja klientrakenduse tehnoloogiate valikut, arendusprotsessi ja testimist. Arendusprotsessi tulemuseks on töötav rakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 6 peatükki, 3 joonist, 3 tabelit.

Abstract

Web Application for Ordering and Selling Food for Fitlap Users

The aim of current thesis is to create a working web application FitCook, where Fitlap users can join to an environment that helps users with cooking. With the application, two parties meet. Those who want food prepared for them according to their own energy needs and those who want to cook.

During the development process, a web application is created that employs home chefs who compile menus of diferent lengths from Fitlap dishes. The person who wants to be cooked for joins the chef he likes and chooses the dishes from the menu he wants. The chef takes into account the person's caloric Intake and accordingly packs the right amount of food into a package that customer can pick up.

The thesis covers the selection of technologies used in web services and client application, the development process and testing. The result of the development process is a working application.

The thesis is in Estonian and contains 24 pages of text, 6 chapters, 3 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* ehk rakendusliides
CLI	*Command-line Interface* ehk käsurea liides
CSS	*Cascading Style Sheets* ehk veebilehtede küljendamisel kasutatav märgistuskeel
DOM	*Domain Object Model* ehk dokumendi objektimudel
DRY	*Don't Repeat Yourself* – disainimuster, mis välistab korduva koodi kirjutamist
HTML	*HyperText Markup Language* ehk veebilehe märgistuskeel
HTTP	*Hypertext Transfer Protocol* ehk protokoll andmete edastamiseks interneti vahendusel
JSON	*JavaScript Object Notation* ehk JavaScript-l põhinev andmevahetusvorming
JWT	*JSON Web Token* ehk JSON-põhine standard ligipääsu tagamiseks
MVC	*Model-View-Controller* ehk mudel-vaade-kontroller-rakendustes kasutatav arhitektuur, mis jaotab rakenduse kolmeks loogiliseks osaks
NPM	*Node Package Manager* ehk JavaScript-i teekide haldaja
ORM	*Object-Relational Mapping* ehk objekt-orienteeritud koodi kaarditamine teise andmevormi
REST	*Representational State Transfer* ehk veebiteenusega suhtlemise liidese arhitektuur
SOAP	*Simple Object Access Protocol* ehk veebiteenusega suhtlemise liidese protokoll
SPA	*Single Page Application* ehk üheleherakendus

SQL	*Structured Query Language* ehk struktureeritud päringukeel
UoW	*Unit of Work* ehk tööühik
Virtual DOM	*Virtual Document Object Model* ehk virtuaalne dokumendi objektide vorm
XML	*Extensible Markup Language* ehk dokumentide märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Inimesed hoolivad oma tervisest järjest rohkem. Kuna kiire elutempo või kogenematuse tõttu ei ole alati aega või tahtmist kodus kokata, siis inimesed langetavad tihti ebatervislikemaid toiduvalikuid.

Fitlap on tervisliku toitumise ja toidukordade planeerimise platvorm, kus inimene saab etteantud retseptide seast endale koostada toitumiskava. Fitlap-i kasutavad inimesed, kes soovivad lisaks tervislikule ja mitmekülgsele toitumisele ka kehakaalu langetada, hoida või tõsta. Fitlap-st leiab iga inimene endale sobiliku toidu, kuna keskkonnas on üle 1000 tervisliku retsepti. Keskkonnaga liitumine on lihtne – tuleb luua konto, sisestada toitumiskava kasutamise eesmärk (kehakaalu tõstmine, hoidmine või langetamine). Teenuse kasutamine on tasuline. Fitlap kasutab spetsiaalset algoritmi toiduportsionite suuruste ja põhiainevahetuse energiavajaduse arvutamiseks ning võtab sisendiks inimese soo, pikkuse, kehakaalu ja aktiivsuse. Eesmärgile lähenedes saab kasutaja sisestada uued andmed ning programm arvutab uue energiavajaduse. Fitlap-i eesmärk on teha tervislik toitumine lihtsamaks ning soodsamaks ning kõige olulisem, et lemmiktoitudest ei pea loobuma.

Käesoleva bakalaureusetöö põhiprobleemiks on inimeste ajapuudusest või kogenematusest Fitlapi-i toitumiskava mitte järgimine. Lõputöös kirjeldatakse ära probleem, vaadeldakse olemasolevaid lahendusi, kirjeldatakse ära loodava rakenduse nõuded ja tehniline lahenduskäik koos põhjendustega. Probleemi lahenduseks pakutakse välja lahendus uue veebirakenduse näol, mis võimaldab kasutajatel lihtsamini tervislikult toituda.

Lõputöö autor kasutab ka ise igapäevaselt Fitlapi juba mitu aastat vormi hoidmiseks. Lisaks vormi hoidmisele on tervis parem ning toidukuludelt kokkuhoid märgatav võrreldes mitte toitumiskava järgides.

1.1 Metoodika

Käesoleva bakalaureusetöö käigus seletatakse kõigepealt probleemi olemust, tuuakse välja olemasolevaid lahendusi koos nende puudustega. Seejärel pakub töö autor välja probleemile sobiva lahenduse.

Lahenduse analüüsi käigus kirjeldatakse ära kasutajagrupid. Erinevate kasutajagruppide kohta koostatakse kasutajalood. Kasutajalood jagunevad funktsionaalseteks ja mittefunktsionaalseteks nõudmisteks.

Töös on kirjeldatud lahenduse valmimist erinevate osadena. Kirjeldatakse veebiteenuse, klientrakenduse ja andmebaasi disaini valmimist ning lahenduse testimist.

2 Ülevaade probleemist

Inimesed, kes tahavad toituda Fitlapi kava järgi, soovivad kiire elutempo või kogenematuse tõttu lasta kellegil teisel teha süüa, et ei peaks loobuma tervislikust toitumisest.

Järgnevalt on välja toodud põhjused, mis takistavad või häirivad Fitlapi kasutajal ise toitumiskava põhjal süüa teha:

- Ajapuudus – Kaari Muldma on oma bakalaureusetöös „Tervisliku toitumise motiivid ja barjäärid“ leidnud, et kõige suurem barjäär tervisliku toitumise juures on ajapuudus [40, joonis 6]. Kiire elutempo tõttu kipub tervislik toitumine jääma tahaplaanile ning selle tulemusena on söömine ebaregulaarne, mis võib põhjustada erinevaid terviseprobleeme.

- Kogenematus – inimene küll soovib tervislikult toituda, kuid puuduliku või vähese toidu valmistamise kogemuse tõttu loobutakse toitumiskava täitmisest.

- Toidujäägid – Fitlapi toitumiskava järgi toitudes on inimesele ette antud toorainete kogused, mida on vaja toidu valmistamiseks kasutada. Kahjuks ei ole võimalik poest osta täpseid koguseid toorainet. Toitumiskava järgijatel tekivad toidujäägid. Mõne inimese jaoks võib see olla tülikas. Alati ei ole võimalik toiduainet säilitada näiteks sügavkülmikus. Halvimal juhul läheb toit vanaks.

2.1 Eksisteerivad lahendused

2.1.1 Söögitellimis platvormid

Ettevõtted nagu Bolt, Wolt ja Telli Toit on teinud toitlustuskohtadest valmistoidu koju tellimise väga lihtsaks. Kiire elutempo tõttu on inimestel võimalik iga päev tellida endale meelepärane toit enda soovitud aadressile. Toidutellimise platvormid pakuvad erinevates kategooriates toite ning samuti leiab ka väga tervislikke menüüsi, mida klient saab endale

tellida. Puudusena saab välja tuua selle, et toidu kogus on alati ühesuurune. Klient ei saa tellida toitu vastavalt tema soovitud toidukorra kaloraažile.

2.1.2 Clean Kitchen

Clean Kitchen on teinud kodus kokkamise lihtsaks. Kasutajal on võimalik valida endale meelepärased söögid retseptikogumikust. Clean Kitchen erineb toidutellimise platvormidest selle poolest, et toidu peab valmis küpsetama toidu tellija. Clean Kitchen pakendab tooraine koos toidu valmistamise retseptidega toidukasti ning teostab kojuveo klendini. Kasutajal jääb ainult toit vastavalt retseptile valmis küpsetada. Iga toidu valmistamise jaoks on toidukastis täpne kogus toorainet ning selletõttu ei kogune külmkappi riknevaid toiduaineid [1]. Puuduseks on paraku jälle olukord, kus kasutaja ei saa ise valida toidu kogust vastavalt tema soovitud toitainete vajadusele.

2.1.3 Valmistoit

Toidupoodides on väga suur valik valmistoitu, mida on lihtne inimesel vajadusel soojendada ning seejärel tarbida. Inimene saab küll vaadata kui palju energiat mingi toit annab ning arvutada palju ta peab valmistoitu ostma, et oma soovitud kaloraaž kätte saada. Kuid see on jällegi tülikas ning toiduainete proportsioonid ei pruugi vastata tervisliku toitumise soovitustele.

2.1.4 SmartFood

SmartFood on keskkond, kus on võimalik valida endale sobiliku kalorite hulgaga pakett. Paketi valik on ainult kindlate kalorite vahemike vahel. Igaks nädalaks on koostatud üks konkreetne menüü, mida on siis võimalik tellida pakettide seast vastavalt soovitud päevasele kaloraažile. Puuduseks on menüü väiksus. Kasutajal ei ole võimalik ise menüüd kokku panna.

2.2 Uue lahenduse skoop

Eksisteerivate lahenduste seas paistab silma kaks põhilist puudust. Esiteks, klient ei saa valida toidu portsioni suurust, et see vastaks tema energiavajadusele. Teiseks, isegi kui saab valmida enam-vähem õige kaloraažiga menüü, siis kliendile pakutakse ainult ühte konkreetset menüüd. Kasutajal ei ole võimalik ise menüüd kokku seada oma toitumiseelistuste põhjal.

Puuduste likvideerimiseks pakub lõputöö autor välja lahenduse uue veebirakenduse loomise näol, et Fitlapi kasutajad saaksid korrektselt täita oma toitumismenüüd. Selleks, et Fitlapi kasutaja saaks osta toiduvalmistamise teenust, selleks peab tal olema kehtiv Fitlapi pakett. Keskkonna kasutamise eest kasutajatelt raha ei võeta. Ainuke tasu on kokale toidu valmistamise eest.

Uue lahenduse loomisega soovitakse teha Fitlapi kasutajate elu mugavamaks, kui kiire elutempo või muude tegurite mõjul on tervisliku söögi valmistamine ja toitumiskava järgimine raskendatud. Bakaraureusetöö autor soovib tulevikus hakata ka raha teenima toiduvalmistamise vahendusteenusega.

3 Loodava veebirakenduse analüüs

Veebirakenduse loomise protsessi juures on väga oluline esmalt ära määrata nõuded, mida tuleb silmas pidada, et kõik rakenduse kasutajad saavad oma soovid rahuldatud. Samuti tuleb valida sobivad komponendid, mille põhjal veebirakendus valmis ehitada.

3.1 Nõuete määramine

Nõuete määramisel võetakse aluseks veebirakenduse kasutajate rollid. Erinevate rollide jaoks koostatakse kasutajalood. Rollideks on toidu tellija, toidu müüja ja administraator. Toidu tellija all peetakse silmas kasutajat, kes soovib, et talle söök valmis küpsetatakse. Toidu müüja on isik, kes valmistab toitusid vastavalt menüüle kasutajatele, kes seda soovivad. Administraatori ülesanne on veebilehte hallata.

3.1.1 Funktsionaalsed nõuded

Toidu tellija funktsionaalsed nõudmised:

- Toidu tellijana soovin sisse logida Fitlapi kasutajakontoga.
- Toidu tellijana soovin, et saaksin tellida toitu vastavalt Fitlapis ettemääratud kaloraažile.
- Toidu tellijana soovin filtreerida menüüsid erinevate parameetrite põhjal.
- Toidu tellijana soovin näha menüüs olevate toitude koostisosi.
- Toidu tellijana soovin näha koka profiili.
- Toidu tellijana soovin, et saaksin menüüst valida kindlad toidukorrad.
- Toidu tellijana soovin näha tellimuste ajalugu.
- Toidu tellijana soovin ühendust võtta kokaga.

- Toidu tellijana soovin anda kommentaari kokale.
- Toidu tellijana soovin näha teiste kasutajate kommentaare kokale.
- Toidu tellijana soovin, et teenuse eest tasumiseks oleks võimalik valida erinevate maksevahendite vahel.

Toidu müüja funktsionaalsed nõudmised:

- Toidu müüjana soovin sisse logida Fitlapi kasutajakontoga.
- Toidu müüjana soovin näha Fitlapi toitusid, mille põhjal koostada menüü.
- Toidu müüjana soovin koostada menüüd.
- Toidu müüjana soovin sisestada toorainete hinnad.
- Toidu müüjana soovin seada aja, millal on viimane aeg toidu tellijatel liituda menüüga.
- Toidu müüjana soovin näha ostukorvi nimekirja toiduainetest.
- Toidu müüjana soovin ostukorvis eemaldada tooted, mis on juba olemas.
- Toidu müüjana soovin, et oleks näha toiduportsjonite kogused iga kliendi kohta.
- Toidu müüjana soovin, et toidu ostjale arvutataks tellimuse hind automaatselt.

Administraatori funktsionaalsed nõudmised:

- Administraatorina soovin hallata kasutajaid.
- Administraatorina soovin hallata teenustasusid.

3.1.2 Mittefunktsionaalsed nõuded

Toidu tellija mittefunktsionaalsed nõudmised:

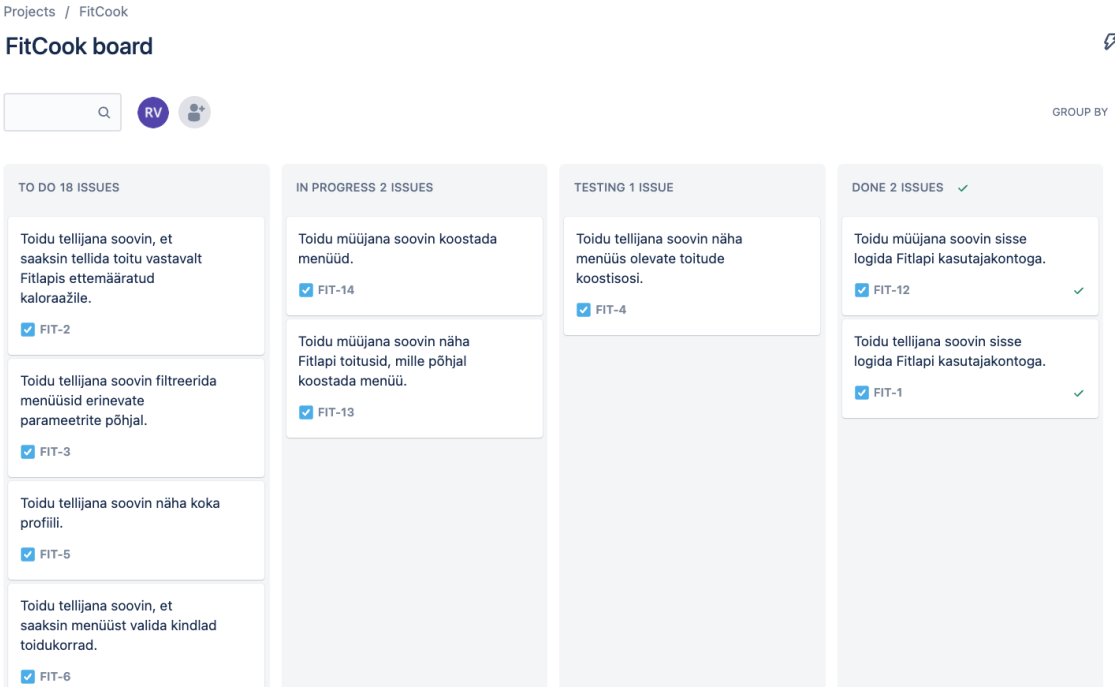
- Toidu tellijana soovin kasutada veebilehte eesti keeles.
- Toidu tellijana soovin kasutada rakendust arvutis kui ka mobiilis.

- Toidu tellijana soovin, et saaksin veebirakendust kasutada ükskõik millises veebibrauseris.

Toidu müüja mittefunktsionaalsed nõudmised:

- Toidu tellijana soovin kasutada veebilehte eesti keeles.
- Toidu tellijana soovin kasutada rakendust arvutis kui ka mobiilis.
- Toidu tellijana soovin, et saaksin veebirakendust kasutada ükskõik millises veebibrauseris.

Arendusprotsessi jälgimiseks on funktsionaalsed kasutajalood kirja pandud Jira arendustööriistas [2]. Jira tarkvara võimaldab sisestada kasutajalood ning liigutada neid arendusprotsessi erinevatesse etappidesse (Joonis 1). Projekti halduseks on olemas teisigi tööriistu, kuid valik langes Jira-le, kuna antud tööriist on üks populaarsemaid, hea dokumentatsiooni ja tugeva toega, kes pidevalt arendab uusi featuure juurde [41].



Joonis 1. Kasutajalood Jira keskkonnas.

3.2 Tehnoloogia valik

Veebirakenduse loomine algab kõigepealt tehnoloogiate valikuga. Tarkvaraarendus on kiiresti muutuv sektor, kus erinevate programmeerimiskeelte ja raamistike populaarsus ja kasutatavus muutub kiiresti ajas. Käesoleva lõputöö raames valib autor need tehnoloogiad, mis on tarkvaraarendajate seas populaarsed ning millega on töö autor ise, kas ülikooli õpingute ajal või tööga seotuna kokku puutunud. Valitud tehnoloogiate valikut kirjeldatakse järgnevates peatükkides.

Veebirakendusi saab ehitada mitmesuguseid arhitektuure järgides. Käesolevas töös luuakse hajus rakendus, mis kasutab klient-server mudelit. Selles mudelis on kaks peamist komponenti – eesrakendus ja tagarakendus. Kolmanda komponendina on kasutuses andmebaas. Klient tähendab antud kontekstis eesrakendust, kus sisu loomiseks kasutatakse HTML-i, CSS-i ja JavaScript-i. Eesrakendus on graafiline liides, millega veebilehe kasutaja suhtleb. Serveri all peetakse silmas tagarakendust, kus on veebirakenduse äriloogika ning mis võtab vastu HTTP päringuid. Andmebaasis hoitakse andmeid, mida oma tööks vajab tagarakendus [25].

Eesrakendus ja tagarakendus peavad omavahel suhtlema. Nii nagu inimesed saavad suhelda omavahel erinevates keeltes, nii ka veebiteenusega on võimalik suhelda mitmel erineval viisil. Veebiteenustel on kasutusel kaks peamist arhitektuuri, SOAP (*Simple Object Access Protocol*) ja REST (*Representational State Transfer*), millega saata andmeid teistesse rakendustesse. SOAP on standardiseeritud protokoll, mis saadab andmeid, kasutades protokolle nagu HTTP ja SMTP. REST on arhitektuuri stiil, mis kasutab andmete liikumiseks HTTP protokolli. REST loodi selleks, et lahendada SOAP-i probleemid. REST-i eeliseks on suurem paindlikkus, kus on võimalik kasutada erinevaid andmeformaate nagu HTML, JSON, XML, kui SOAP-i puhul on võimalik kasutada ainult XML formaati. Veebiteenused, mis kasutavad REST arhitektuuri on parem jõudlus. Tänapäeval on REST populaarseim valik arendajate seas [26].

Lõputöö projektis võetakse kasutusele REST lähenemine, kuna see on hetkel populaarseim valik arendajate seas ja lõputöö autor puutub REST arhitektuuriga igapäevatöös kokku.

Kuna bakalaureusetöö raames loodav veebirakendus ehitatakse klient-server mudeli põhjal, siis alljärgnevalt analüüsitakse kliendi- ja teenusepoolseid tehnoloogiaid. Valikute

tegemisel võetakse arvesse, kas tehnoloogia kasutamise eest tuleb raha maksta, mis on töö autori kogemus tehnoloogiatega ja kui hästi tehnoloogiad omavahel kokku sobivad.

3.2.1 Teenusepoolse tehnoloogia valik

Serveripoolsed programmeerimisraamistikud ja programmeerimiskeeled käivad käsikäes. See tähendab, et ühte kindlat programmeerimiskeelt ei saa kasutada kõikides raamistikes. Igale raamistikule on mõeldud kindlad programmeerimiskeeled. Järgnevalt on välja toodud serveripoolsed programmeerimisraamistikud ja programmeerimiskeeled, mis sobivad loodava rakenduse ehitamiseks.

Programmeerimiskeeled:

- C# - C# on kaasaegne, objektorienteeritud ja tüübikindel programmeerimiskeel. C# võimaldab arendajatel ehitada turvalisi ja keerulisi rakendusi, mis töötavad .NET raamistikus [5].

- PHP – PHP on serveripoolne programmeerimiskeel, mis sobib dünaamiliste veebilehtede arenduseks. Samuti on võimalik PHP-d kasutada ka skriptimiskeelena. Kui veebibrauserist tehakse HTTP päring serverile, siis server töötleb PHP koodi, et genereerida HTML dokument, mis siis saadetakse kliendile tagasi [4] . PHP-d peetakse üheks populaarsemaks keeleks. Peaaegu 80% kõikidest veebilehtedest kasutab PHP-d [29].

- Java – Java on objektorienteeritud, klassipõhine programmeerimiskeel, mida saab kasutada kõikidel platvormidel [27]. Java-t peetakse üheks kiiremaks, turvalisemaks ja usaldusväärsemaks programmeerimiskeeleks, mida kasutatakse nii väikestes kui ka suurtes arendustes [28].

- JavaScript – JavaScript on programmeerimiskeel, mis võimaldab luua veebilehele dünaamilist sisu. JavaScripti saab kasutada nii klientrakenduse arendamiseks kui ka serveripoolse rakenduse arendamiseks [3].

Raamistikud:

- .NET – .NET on Microsofti poolt loodud vabavaraline arendusplatvorm, mis võimaldab ehitada erinevaid rakendusi. .NET rakendusi saab kirjutada C#, F# ja Visual Basic programmeerimiskeeles [6].

- Laravel – Laravel on kõige populaarsem tasuta saadaval avatud lähtekoodiga PHP raamistik. Laravel hõlbustab kõige paremini rakenduse arendamist. Tema eeliseks teiste PHP raamistike ees turvalisuse tagamine, lisaks lihtsustab arendusprotsessi autentimise, vahemälu kasutamisega, sessiooni haldusega ja võrguliikluse haldamisega [29].

- Spring Framework – Spring on üks populaarsemaid Java raamistike veebirakenduste loomiseks. Spring-i raamistik võimaldab teha Java veebirakendused turvaliseks, luua ühendus andmebaasiga ning arendada veebiteenuseid[30].

- ExpressJS – JavaScripti saab kasutada serveripoolses rakenduses, kui kasutada Node.js keskkonda. ExpressJS on üks parimaid Node.js raamistikke, mis lihtsustab Node.js rakenduste loomist tänu erinevate teekide olemasolule. ExpressJS ei nõua palju aega selle õppimiseks, kui põhiteadmised Node.js on olemas [31].

Järgnevas tabelis on välja toodud serveripoolse programmeerimiskeelte võrdlus, kus kajastatakse töö autori kogemus ning kui keeruline on programmeerimiskeelt õppida ja milline on toetav kogukond (Tabel 1). Kõikidel allpool nimetatud keeltel on olemas hea dokumentatsioon ja suur kasutajaskond.

Tabel 1. Teenusepoolsete programmeerimiskeelte võrdlus

Keel	**Kogemus**	**Õppimiskeerukus**	**Toetav kogukond**
C#	Väga hea	Keskmine [43]	Väga suur [44]
PHP	Kehv	Madal [19]	Suur [45]
Java	Rahuldav	Keskmine [7]	Väga suur [46]
JavaScript	Hea	Madal [32]	Väga suur [47]

Lõputöö kirjutamiseks ja rakenduse arendamiseks on piiratud aeg ning uue programmeerimiskeele õppimine võtaks liiga palju aega. Mõistlik oleks valida need keeled, mida lõputöö autor oskab kõige rohkem. Nendeks keelteks on C# ja Java.

Lõputöö autor teeb valiku C# kasuks, kuna puutub antud keelega igapäevaselt kokku ning mille õppimiskeerukus on keskmine. Lisaks on C#-l väga suur toetav kogukond, programmeerimiskeelt pidevalt arendatakse ning probleemidele leiab kiirelt lahenduse tänu suurele toele [44]. Lõputöö autoril on töökogemust C#-ga 1,5 aastat. Java programmeerimiskeelt on töö autor kasutanud ainult kuu aega praktikal olles.

Eelpool sai mainitud, et raamistikud käivad programmeerimiskeeltega käsikäes. Sellest tulenevalt saab serveripoolsetest raamistikest valida .NET raamistiku.

3.2.2 Klientrakenduse tehnoloogia valik

Kuna serveripoolne lahendus on REST API näol, siis klientrakendus peab oskama teha päringuid veebiteenuse pihta. Veebiteenus annab andmeid välja JSON kujul. Sobilik lahendus antud juhul on kasutada JavaScript-i raamistikke, mis oskavad suhelda serveripoolse rakendusega, töödelda andmeid ning kuvada kasutajale infot graafilise kasutajaliidese kaudu. Selleks, et kasutajale graafilist pilti kuvada, tuleb kasutada ka HTML-i ja CSS-i. HTML-i ja CSS-i näol on tegemist keeltega, mis vastavalt aitavad struktureerida veebilehte ja teha veebileht ilusaks [8].

Selleks, et klientrakendus oleks kaasaegne, pakuks kasutajale head kasutajakogemust, võimaldaks kiiret lehe laadimist, siis on vaja vaadata raamistikke, millega on võimalik ehitada SPA (*Single Page Application*) ehk üheleherakendusi. SPA rakendusel on ainult üks veebileht. Kui veebileht saadab päringu veebilehe serverile, siis SPA saadab tagasi HTML faili koos JavaScripti ja CSS koodiga. Niipea kui veebibrauser käivitab JavaScripti failid, siis server pärib andmed ja veebilehe sisu muudetakse dünaamiliselt, ilma, et peaks veebilehte uuesti laadima [21].

Järgnevalt on välja toodud suuremad ja populaarsemad JavaScripti raamistikud, millega on võimalik luua üheleherakendust [21] [20]:

- React – React on kõige populaarsem JavaScripti raamistik klientrakenduste loomisel. React on loodud, et ehitada kiireid ja interatiivseid kasutajaliideseid veebi- ja mobiilirakenduste jaoks. Tegemist on vabavaralise, komponendipõhise raamistikuga. React kasutab Virtual DOM-i. See tähendab, et kui mingis komponendis toimus muudatus, siis muudetakse ainult seda osa HTML-s, kus toimus muudatus [33].

- Angular – Angular on arendusplatvorm, mis on ehitatud TypeScripti-ga. Angular on komponendipõhine raamistik, millega saab ehitada skaleeritavaid kasutajaliideseid [9]. TypeScript on tugevalt tüübitud programmeerimiskeel, mis kompileerib JavaScriptiks [10].

- Vue.js – Vue on progressiivne raamistik kasutajaliideste ja üheleherakenduste ehitamiseks. Vue põhiteek on mõeldud ainult vaadete kihile, kuid väga lihtne on lisada teisi JavaScripti teeke, mis teeb temast võimeka raamistiku [11].

Kõik eelpool mainitud raamistikud sobivad üheleherakenduste arendamiseks. Järgnevas tabelis (Tabel 2) tuuakse välja klientrakenduse raamistike võrdlus. Võrreldakse erinevate raamistike õppimiskeerukust, töö autori kogemust vastava raamistikuga ning oluline on ka toetav kogukond.

Tabel 2. Klientrakenduse raamistike võrdlus

Raamistik	**Kogemus**	**Õppimiskeerukus**	**Toetav kogukond**
React	Puudub	Madal [34]	Suur [48]
Angular	Kehv	Kõrge [34]	Väga suur [48]
Vue.js	Hea	Madal [34]	Keskmine [48]

Kõikide eelnimetatud raamistike õppimiseks on olemas väga hea dokumentatsioon. Kõiki neid raamistike saab kasutada lõputöö projekti ehitamiseks. Angulari raamistikus on kõige rohkem funktsionaalsust, et ehitada suuri ja keerulisi veebirakendusi. Angulari õppimine nõuab tunduvalt rohkem aega ja selles raamistikus ehitatud veebirakendused on aeglasemad kui teised raamistikud [34]. Lõputöö käigus arendatav rakendus ei ole suuremahuline, siis sobivad React ja Vue.js paremini. React on küll kõige populaarsem raamistik, kuid lõputöö raames tuleb arvestada ka ajalise mahuga, mis kuluks uue raamistiku õppimiseks. Lõputöö autoril on kõige suurem kogemus Vue raamistikuga, mida ta kasutab igapäevatöös. Rakenduses nõutud funktsionaalsuse saavutamiseks saab lisada Vue raamistikku teisi teeke. Kliendipoolseks tehnoloogiaks valitakse Vue.

3.3 Andmebaasi valik

Andmete salvestamiseks on vaja andmebaasi. Andmebaasid jagunevad kahte gruppi. Nendeks on relatsiooniline ehk SQL andmebaas ja mitterelatsiooniline ehk NoSQL andmebaas. Relatsioonilises andmebaasis salvestatakse andmed tabelitesse kirjetena. Kirjed paiknevad tabelis ridadena. Tabelid seotakse omavahel unikaalsete võtmete abil. Mitterelatsioonilises andmebaasis ei ole tabeleid. Lõputöö autoril on kogemus ainult relatsiooniliste andmebaasidega ning sellest lähtuvalt võetakse vaatluse alla ainult tuntumad relatsioonilised andmebaasid. Relatsioonilisest andmebaasist info pärimiseks kasutatakse SQL (*Structured Querying Language*) päringukeelt. SQL-ga on võimalik luua, pärida, uuendada ja kustutada kirjeid andmebaasi tabelitest. Järgnevalt on välja toodud mõned tuntumad SQL andmebaasid [35].

- Microsoft SQL Server – Microsoft SQL Server on relatsiooniline andmebaasi haldussüsteem, mis on arendatud Microsoft-i poolt. Lisaks SQL-i kasutamisele, on Microsoft lisanud enda implementatsiooni SQL-st, mida kutsutakse Transact-SQL-ks. Esialgu töötas ainult Windowsi platvormil, kuid aastal 2016 tehti see võimalikuks ka Linux-i platvormil [12].

- MySQL - MySQL on vabavaraline realtsiooniline andmebaas, mis on arendatud Oracle poolt. MySQL-i peetakse kõige populaarsemaks andmebaasiks tänu tema kiirusele, jõudlikusele ja turvalisele andmehaldussüsteemile [13].

- Oracle Database - Oracle Database on relatsiooniline andmebaas, mis kuulub Oracle-le. Arendatud juba aastal 1977. Oracle andmebaasi kasutavad üldjuhul suured ettevõtted, kuna nende andmebaas võimaldab hoiustada suures koguses andmeid ning andmetele saab ligi kiirelt [36].

- PostgreSQL – PostgreSQL on võimas, avatud lähtekoodiga objekt-relatsiooniline andmebaas, mis on tuntud oma usaldatavuse, funktsionaalsuste rohkusega ja jõudlusega [37].

Tabel 3. Relatsiooniliste andmebaaside võrdlus

Andmebaas	**Kogemus**	**Maksumus**
Microsoft SQL Server	Hea	Tasuline [35].
MySQL	Halb	Tasuta [35].
Oracle Database	Puudub	Tasuline [35].
PostgreSQL	Hea	Tasuta [35].

Lõputöö autor kasutab oma igapäevatöös Microsoft SQL Server andmebaasi. Lõputöö projekt ei näe ette, et veebirakenduses kasutavate tehnoloogiate peale peaks raha kulutama. Rahalise kulu tõttu jäävad Microsoft SQL Server ja Oracle andmebaas valikust välja. Valikusse jääb järgi MySQL ja PostgreSQL. Õpingute ajal puutus tudeng kokku PostgreSQL andmebaasiga ning tulenevalt suuremast kogemusbaasist valitakse lõputöö projekti andmebaasiks PostgreSQL.

4 Veebirakenduse arendus

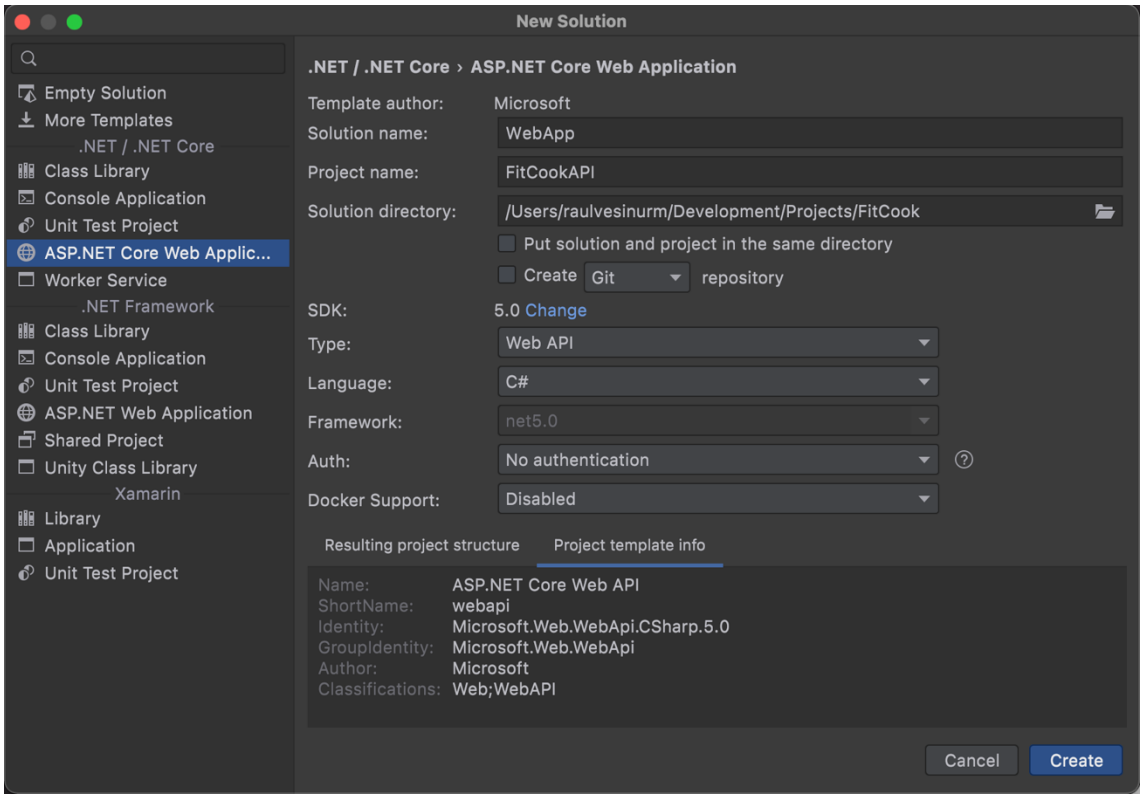
Veebirakenduse arendus on jaotatud kolme peatükki: veebiteenuse arendus, klientrakenduse arendus ja testimine. Veebirakenduse arendamiseks valitud tehnoloogiad on kirjeldatud loodava veebirakenduse analüüsi peatükis.

4.1 Veebiteenuse arendus

Veebirakenduse veebiteenusepoolse tehnoloogia valik langes .NET raamistiku kasuks, mis kasutab C# programmeerimiskeelt. Lõputöö projekti teostamiseks kasutatakse .NET 5.0 versiooni. Veebiteenus on alus klientrakendusele. Veebiteenus peab suutma vastu võtma HTTP päringuid, vastutab kogu rakenduse äriloogika eest. Läbi HTTP päringute käib andmete vahetus, mis tähendab ,et veebiteenusel peab olema tugi andmebaasist andmete lugemisel, sinna kirjutamisel, muutmisel ja kustutamisel. Veebirakenduse ehitamiseks ja käivitamiseks kasutatakse macOS operatsioonisüsteemi.

4.1.1 ASP.NET Core rakenduse loomine

.NET raamistikus veebirakenduse loomiseks on vaja valida ASP.NET Core tehnoloogia. Projekt luuakse JetBrains Rider arenduskeskkonnas. Lisaks JetBrains Rider-le on veel mitmeid populaarseid tööriistu projekti arendamiseks [42]. Töö autor valis JetBrains Rider-i, kuna antud tööriista saab kasutada tasuta tänu ülikoolile, lisaks on olemas toetav kogukond ja kogemus antud tarkvaraga. Järgnevalt jooniselt (Joonis 2) on näha rakenduse loomise protsessi:



Joonis 2. ASP.NET Core veebiteenuse loomine

Veebiteenuse nimeks on pandud FitCookAPI ja veebirakenduse tüübiks on valitud Web API. Kuna teenus tegeleb ainult info väljastamise ja salvestamisega, siis ei ole vaja luua MVC (Model-View-Controller) mustriga veebirakendust. Veebiteenusel puudub View komponent.

4.1.2 Veebiteenuse osad

Veebiteenuse projekt on jagatud kihtideks. Iga kiht vastutab kindla töö eest. Projekti loomisega peatükis 4.1.2 loodi kohe kontrolleri kiht. Järgnevalt räägitakse lähemalt projektis kasutatavatest kihtidest.

Selleks, et veebiteenus saaks andmeid tagastada ja salvestada, on vaja andmebaasi. Projektis kasutatakse Code First ehk kood enne lähenemist [22]. Code First lähenemise korral luuakse kõigepealt domeenimudel ehk C# klassid. Domeenimudeli põhjal genereerib Entity Framework Core andembaasi. Domeenimudelit kujutamiseks on tehtud Domain projekt.

Andmetele juurdepääsu jaoks on tehtud DAL (*Data Access Layer*) kiht. Kasutusele on võetud repositooriumi (Respository pattern) ja UoW (*Unit of Work*) muster [23]. Selleks,

et andmebaasi tabelist andmeid küsida ja muuta, tuleb selleks teha meetodid. Üldjuhul on andmebaasis rohkem kui üks tabel. Kui iga tabeli kohta hakata samasuguseid meetodeid kirjutama, kus ainult tabeli nimetus erineb, siis rikutakse DRY (*Don't Repeat Yourself*) printriipi. Repositooriumi mustrit kasutatakse selleks, et luua üldine repositooriumi klass, kus on realiseeritud andmete pärimise loogika. Kõik ülejäänud repositooriumid saavad laiendada enda klassi üldise repositooriumiga ja kasutada sealset funktsionaalsust andmete pärimiseks andmebaasist. Repositooriume kasutatakse äriloogika kihis. Mugavamaks kasutamiseks luuakse eraldi klass (UoW), kus on kõik repositooriumid kättesaadavad.

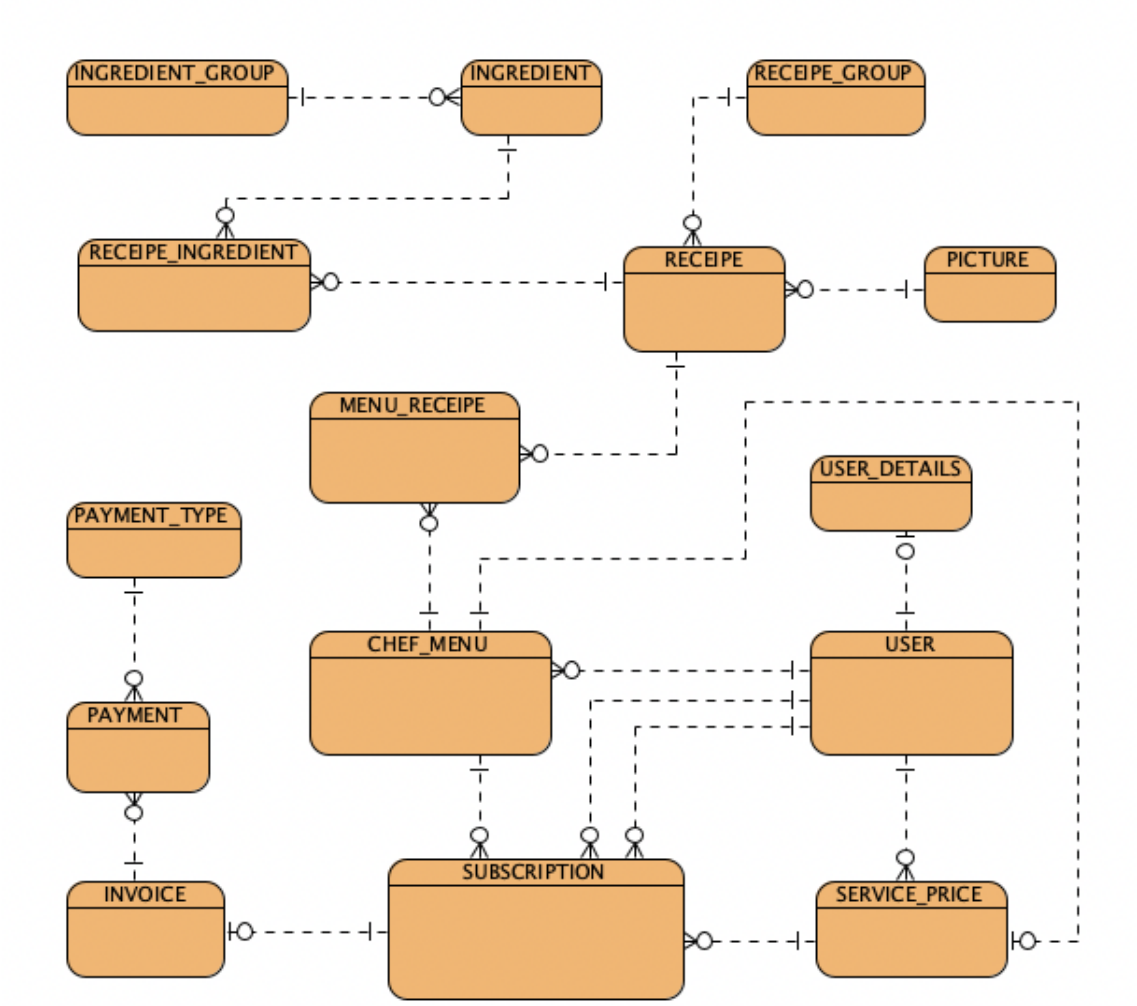
Äriloogika teostamiseks on loodud BLL (*Business Logic Layer*) ehk äriloogika kiht. Äriloogika kiht saab andmed DAL kihi käest. Seejärel tehakse andmetega erinevaid äriloogilisi muudatusi, mis siis küsimise peale antakse kontrolleritele.

Selleks, et klientrakendus saaks suhelda veebiteenusega, on vaja kontrollereid. Kontrollerid asuvad WebApp projektis. Kui klientrakendus pöördub HTTP päringuga kontrolleri poole ja soovib saada andmeid, siis kontroller küsib soovitud andmed äriloogika kihist.

Alati ei pruugi kõik veebiteenuse kontrollerid olla kättesaadavad kõigile. Näiteks veebirakenduse avaleht, sisselogimiseleht ja registreerimiseleht on avalikud. Neile pääsevad kõik ligi. Kuid on ka kontrollereid, mis nõuavad autoriseerimist, näiteks kasutaja profiililileht. Veebiteenuses on kasutatud autoriseerimiseks JWT-d (*Json Web Token)*, mis on avatud standard turvaliseks andmete vahetamiseks JSON objekti kaudu [14]. Kui kasutaja logib veebirakendusse sisse, siis genereeritakse talle krüpteeritud võti, mis pannakse iga HTTP päringuga veebiteenuse poole kaasa. Autoriseerimist nõudev kontroller tuvastab JWT ja lubab ligipääsu andmetele.

4.2 Andmebaasi mudel ja selle ühendamine veebiteenusega

Andmete hoiustamiseks on vaja andmebaasi. Andmebaas koosneb tabelitest ja nendevahelistest seostest. Andmebaasi koostamise aluseks on lõputöös käsitletav probleem ja rakendusele pandud funktsionaalsed nõuded. Andmebaasimudel on koostatud Visual Paradigm [18] tööriista abil. Joonisel 3 on näidatud olemi-suhte diagramm.



Joonis 3. Lihtsustatud andmebaasi skeem

Andmetele ligipääs on väga oluline igas tarkvara rakenduses. ASP.NET Core pakub erinevaid võimalusi andmebaasiga ühendamiseks [24]. ASP.NET Core rakenduses on soovitatud kasutada Entity Framework Core (EF Core) tööriista, kui soovitakse töötada relatsiooniliste andmetega. EF Core pakub ORM (Object-Relational Mapper) funktsionaalsust, mis võimaldab arendajatel töötada objektidega.

Analüüsi käigus valiti välja PostgreSQL andmebaas. PostgreSQL-i ühendamine rakendusse ei ole keeruline. Selleks, et kasutada EF Core koos PostgreSQL andmebaasiga, on vaja lisada NuGet paketihalduris Npgsql.EntityFrameworkCore.PostgreSQL teek [38].

Andmebaasimudeli põhjal loodi domeeni kihis C# klassid. Klassid kujutavad endast tabeleid ning klassides olevad väljad tabeli väljasid. Klasside põhjal loodud objektid

kujutavad endast kirjeid andmebaasi tabelis. Selleks, et EF Core teaks, milliseid andmebaasitabeleid on vaja luua, tuleb teha eraldi klass andmebaasi konteksti kohta, kus öeldakse, millistest klassidest ehitada tabelid. Peale andmebaasi konteksti loomist tuleb genereerida migratsioon, et luua andmebaas. Käsureale sisestatakse järgnev käsk: dotnet ef migrations add <migratsiooni nimi>. Käsuga luuakse Migrations kaust, mis sisaldab migratsiooni faile. Ainult migratsioonifailide loomisest ei piisa, et andmebaasi luua. Tuleb käivitada käsureal järgmine käsk: dotnet ef database update, mis genereeritud migratsioonifailide põhjal loob andmebaasi.

4.3 Klientrakenduse lahendus

Klientrakendus on iseseisev rakendus veebiteenusest, mida hakkavad veebirakenduse kasutajad kasutama. Tänapäevase modernse kasutajaliidese ehitamiseks kasutatakse SPA (Single Page Application) lähenemist, mille tööga saab suurepäraselt hakkama Vue raamistik. Vue raamistik valiti analüüsi osas klientrakenduse loomise tehnoloogiaks.

Kõigepealt on vaja arvutisse paigaldada NPM (Node Package Manager), et Vue rakenduse saaks luua. NPM abil hallatakse teeke ja käivitatakse klientrakenduse server [15]. Vue rakenduse loomiseks tuleb installeerida NPM abil Vue CLI . Vue CLI on käsurea tööriist, millega saab Vue rakendust luua. Vue CLI installeerimiseks sisestame käsureale järgneva käsu: npm update -g @vue/cli. Projekti loomiseks sisestame käsureale käsu vue create <projekti nimi>, kus tuleb seadistada projekt. Peale installeerimise protsessi saab rakenduse käivitada käsuga npm run serve ning rakendust on võimalik näha aadressil http://localhost:8080 [39].

Vue projektis kasutatakse JavaScripti asemel TypeScript programmeerimiskeelt. TypeScript on tugevalt tüübitud programmeerimiskeel. TypeScript on laiend JavaScriptile, mis kompileerub ikkagi JavaScript-ks.

Vue kasutab vaadete loomiseks komponente, mis on .vue laiendiga failid. Vue komponendid on korduvkasutatavad ning koosnevad HTML-i, CSS-i ja JavaScripti koodist [16].

Serveripoolse rakendusega suhtlemiseks on loodud klientrakenduse projektis teenus, mis võimaldab suhelda veebiteenusega. API-ga suhtlemiseks on kasutatud axious teeki, mis

tuleb installeerida käsuga npm install axious. Axious võimaldab teha HTTP päringuid andmete pärimiseks ja andmete saatmiseks [17].

Veebiteenuse päringute autentimiseks on kasutatud JWT. Kui veebirakenduse kasutaja logib keskkonda sisse, serveripoolne rakendus genereerib JWT ning saadab päringuga klientrakendusele. Klientrakendus peab talletama JWT ning iga päringuga veebiteenuse poole JWT kaasa andma.

4.4 Testimine

Rakenduse loomisel on olulisel kohal testimine. Ilma testimiseta ei saa garanteerida, kas rakenduse funktsionaalsus töötab ootuspäraselt. Projekti testimiseks kasutatakse automaatteste. Kõige suurem osa teste on kirjutatud Unit testidena ehk üksustestidena. Lisaks üksustestidele kasutatakse ka integratsiooniteste.

Üksustestid tagavad, et rakenduse individuaalsed komponendid töötavad nii nagu peavad. Integratsioonitestidega testitakse, kuidas mitu komponenti omavahel töötavad. Rakenduses testitakse peamiselt integratsioonitestidega veebiteenuse ja andmebaasi vahelist suhtlust.

5 Hinnang loodud veebirakendusele

Lõputöö raames loodud veebirakendust saab hinnata saavutatud funktsionaalsuse alusel, tehnoloogia valiku modernsuse põhjal ning veebirakenduse edasiarendamise osas.

5.1 Saavutatud kasutatavus

Kõik mittefunktsionaalsed ja funktsionaalsed nõuded, mida sai jälgitud ka Jira keskkonnas, said lõputöö raames valmis.

Hetkel ei ole võimalik loodud veebirakendust kasutada reaalsete Fitlapi kasutajate peal, kuna liidestus Fitlap-ga ei ole tehtud. Liidestus Fitlap-ga ei mahtunud lõputöö skoopi.

Veebirakendus on kasutatav viisil, kus kasutajad saavad sisse logida. Hetkel on tehtud demo kontod, mis on andmebaasi salvestatud. Sisse logides saavad kasutajad otsida menüüsid, mida on teised Fitlapi kasutajad koostanud. Need, kes on huvitatud toidu valmistamisest, saavad koostada menüüsid, millega saavad siis toidu tellijad liituda. Toidu valmistaja näeb, kes on liitunud tema menüüga, välja arvutatud toiduainete kogused ostukorvi jaoks ning kui palju toitu ta peab igale toidu tellijale välja kaaluma.

Töö autor tegeleb peale lõputöö valmimist projekti arendusega edasi, et teha ka liidestus Fitlapi-ga. Seejärel on võimalik veebirakendust kasutada reaalsete Fitlapi kasutajate peal.

5.2 Kasutatavad tehnoloogiad

Projekti arendustöös kasutati ainult enamlevinuid tehnoloogiaid, millel on olemas hea dokumentatsioon ja suur kasutajabaas. Suurt rõhku mängis tehnoloogiate valikul ka lõputöö autori kogemus antud tehnoloogiates.

Veebiteenuse arendamiseks kasutati .NET keskkonna ASP.NET Core raamistikku versiooniga 5.0. Klientrakenduse arenduses kasutati JavaScripti raamistikku Vue.js.

Veebirakenduses kasutatud tehnoloogiaid saab kõrgelt hinnata nende aktuaalsuse ja sobivuse pärast.

5.3 Võimalused edasi arenduseks

Lõputöös loodava veebirakendusega liitumise nõudeks on, et kasutajal peab olema kehtiv Fitlapi kasutajakonto. Lõputöö skoopi ei mahtunud liidestus Fitlapi veebiteenusega. Üks võimalik edasi arendus oleks võtta ühendust Fitlapi meeskonnaga ning rääkida neile lõputöös loodud rakendusest ning kas Fitlap oleks nõus tegema koostööd. Siis saaks sünkroniseerida andmeid Fitlapi ja lõputöös loodud rakenduse vahel. Verifitseerida kasutajaid ning saada infot kasutajate energiavajaduse kohta.

Teiseks edasiarenduse ideeks oleks lisada transpordiviisis toidu koju veoks. Hetkel on nii, et kasutaja peab ise söögile järgi minema.

6 Kokkuvõte

Antud bakalaureusetöö eesmärgiks oli luua veebirakendus, mis abistaks Fitlapi kasutajaid toidu valmistamisega ning mis samal ajal täidaks ka eksisteerivate lahenduste puudujääke. Loodud veebirakenduse abil saavad Fitlapi kasutajad liituda keskkonnaga, kus kasutajad saavad valida endale meelepärase menüü, mis on kellegi teise Fitlapi kasutaja poolt loodud, ning sealjuures arvestab inimese päevast energiavajadust.

Lõputöö raames ei loodud mobiilirakendust, kuna kasutajaliides on tehtud ka mobiiltelefonide jaoks sobivaks. Lisaks ei võta veebirakendus mobiiltelefonides mälu, mida mobiilirakendus teeks.

Loodava veebirakenduse analüüsi peatükk andis hea ülevaate planeeritud funktsionaalsustest, veebiteenuse ja klientrakenduse tehnoloogiate valikust ning andmebaasi valikust. Veebirakenduse arenduse peatükk annab ülevaate projekti arenduskäigust, kus kirjeldatakse veebiteenuse ja klientrakenduse arhitektuuri, turvalisust ning testimist.

Lõputöö projekti saab lugeda õnnestunuks, kuna kõik analüüsi peatükis kirja pandud nõuded said täidetud. Loodud rakendus võimaldab Fitlapi kasutajatel jätkata tervisliku toitumisega ilma, et nad peaksid aja puuduse, kogenematuse või mingil muul põhjusel loobuma Fitlapi kava jälgimisest.

Kasutatud kirjandus

[1] „Clean Kitchen,“ Clean Kitchen, [Võrgumaterjal]. Available: https://www.cleankitchen.ee/pages/kuidas-see-tootab. [Kasutatud 6 11 2021].

[2] „Jira,“ Atlassian, [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira. [Kasutatud 6 11 2021].

[3] „What is JavaScript?,“ MDN Web Docs, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 6 11 2021].

[4] „What is PHP,“ PHP Tutorial, [Võrgumaterjal]. Available: https://www.phptutorial.net/php-tutorial/what-is-php/. [Kasutatud 6 11 2021].

[5] „A tour of the C# language,“ Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 6 11 2021].

[6] „What is .NET Framework?,“ Microsoft, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework. [Kasutatud 6 11 2021].

[7] „Is Java Hard To Learn For A Beginner? The Hard Facts,“ Comp Sci Central, [Võrgumaterjal]. Available: https://compscicentral.com/is-java-hard-to-learn/. [Kasutatud 6 11 2021].

[8] „HTML & CSS,“ w2.org, [Võrgumaterjal]. Available: https://www.w3.org/standards/webdesign/htmlcss. [Kasutatud 6 11 2021].

[9] „What is Angular?,“ Angular, [Võrgumaterjal]. Available: https://angular.io/guide/what-is-angular. [Kasutatud 20 11 2021].

[10] „What is TypeScript?,“ TypeScript, [Võrgumaterjal]. Available: https://www.typescriptlang.org/. [Kasutatud 20 11 2021].

[11] „Introduction,“ Vue.js, [Võrgumaterjal]. Available: https://vuejs.org/v2/guide/. [Kasutatud 20 11 2021].

[12] „What is SQL Server,“ sqlservertutorial.net, [Võrgumaterjal]. Available: https://www.sqlservertutorial.net/getting-started/what-is-sql-server/. [Kasutatud 20 11 2021].

[13] „What is MySQL? Everything You Need to Know,“ Talend, [Võrgumaterjal]. Available: https://www.talend.com/resources/what-is-mysql/. [Kasutatud 20 11 2021].

[14] „Introduction to JSON Web Tokens,“ JWT, [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 20 11 2021].

[15] „About npm,“ npmjs, [Võrgumaterjal]. Available: https://docs.npmjs.com/about-npm. [Kasutatud 20 11 2021].

[16] „Components Basics,“ Vue.js, [Võrgumaterjal]. Available: https://v3.vuejs.org/guide/component-basics.html#base-example. [Kasutatud 20 11 2021].

[17] „axios,“ npm, [Võrgumaterjal]. Available: https://www.npmjs.com/package/axios. [Kasutatud 20 11 2021].

[18] „Design Database with Professional ERD Software,“ Visual Paradigm, [Võrgumaterjal]. Available: https://www.visual-paradigm.com/features/database-design-with-erd-tools/. [Kasutatud 6 11 2021].

[19] „How to Learn PHP,“ Career Karma, 2 11 2021. [Võrgumaterjal]. Available: https://careerkarma.com/blog/how-to-learn-php/. [Kasutatud 6 11 2021].

[20] „Front end Frameworks for Web Development in 2021,“ Communication Crafts, 4 8 2021. [Võrgumaterjal]. Available: https://www.communicationcrafts.com/frontend-frameworks-for-web-development-in-2021/. [Kasutatud 6 11 2021].

[21] „5 Best Single Page Application Frameworks To Consider For Web Apps,“ Monocubed, 1 11 2021. [Võrgumaterjal]. Available: https://www.monocubed.com/top-single-page-application-frameworks/. [Kasutatud 20 11 2021].

[22] „Tutorial: Code First Approach in ASP.NET Core MVC with EF,“ Medium, 2 4 2021. [Võrgumaterjal]. Available: https://medium.com/c-sharp-progarmming/tutorial-code-first-approach-in-asp-net-core-mvc-with-ef-5baf5af696e9. [Kasutatud 20 11 2021].

[23] „https://medium.com/c-sharp-progarmming/repository-pattern-and-unit-of-work-with-asp-net-core-5-60f5df3e9dea,“ Medium, 16 7 2021. [Võrgumaterjal]. Available: https://medium.com/c-sharp-progarmming/repository-pattern-and-unit-of-work-with-asp-net-core-5-60f5df3e9dea. [Kasutatud 20 11 2021].

[24] „Working with Data in ASP.NET Core Apps,“ Microsoft, 18 11 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/work-with-data-in-asp-net-core-apps. [Kasutatud 20 11 2021].

[25] H. Dhaduk, „An Ultimate Guide of Web Application Architecture,“ Simform, 31 5 2021. [Võrgumaterjal]. Available: https://www.simform.com/blog/web-application-architecture/. [Kasutatud 6 11 2021].

[26] A. Monus, „SOAP vs REST vs JSON - a 2021 comparison,“ Raygun, 5 5 2021. [Võrgumaterjal]. Available: https://raygun.com/blog/soap-vs-rest-vs-json/. [Kasutatud 6 11 2021].

[27] A. Johari, „What Is Java? A Beginner’s Guide to Java and Its Evolution,“ Edureka, 30 9 2021. [Võrgumaterjal]. Available: https://www.edureka.co/blog/what-is-java/. [Kasutatud 6 11 2021].

[28] J. Hartman, „What is Java? Definition, Meaning & Features of Java Platforms,“ Guru99, 7 10 2021. [Võrgumaterjal]. Available: https://www.guru99.com/java-platform.html. [Kasutatud 6 11 2021].

[29] A. Njenga, „10 Popular PHP frameworks for web developers to consider in 2021,“ Raygun, 21 2 2021. [Võrgumaterjal]. Available: https://raygun.com/blog/top-php-frameworks/. [Kasutatud 6 11 2021].

[30] M. WARREN, „The 10 Most Popular Java Frameworks,“ BairesDev, 4 10 2021. [Võrgumaterjal]. Available: https://www.bairesdev.com/blog/the-10-most-popular-java-frameworks/. [Kasutatud 6 11 2021].

[31] T. Kaneriya, „12 Best Nodejs Frameworks for Web Apps in 2022,“ Simform, 5 1 2021. [Võrgumaterjal]. Available: https://www.simform.com/blog/best-nodejs-frameworks/. [Kasutatud 6 11 2021].

[32] A. Meyster, „How long does it take to learn JavaScript?,“ Jaxenter, 15 5 2020. [Võrgumaterjal]. Available: https://jaxenter.com/learn-javascript-171411.html. [Kasutatud 6 11 2021].

[33] T. Sufiyan, „What is React: Definition, Why ReactJS, its Features and Installation,“ Simplilearn, 15 11 2021. [Võrgumaterjal]. Available: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs. [Kasutatud 20 11 2021].

[34] K. Samson, „Which JavaScript Framework Should you learn in 2021?,“ Dev.to, 17 2 2021. [Võrgumaterjal]. Available: https://dev.to/kalashin1/which-javascript-framework-should-you-learn-in-2021-27lj. [Kasutatud 20 11 2021].

[35] T. Pattinson, „Relational vs. non-relational databases,“ Pluralsight, 13 8 2020. [Võrgumaterjal]. Available: https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases. [Kasutatud 20 11 2021].

[36] J. Meah, „Oracle Database,“ Techopedia, 21 06 2021. [Võrgumaterjal]. Available: https://www.techopedia.com/definition/8711/oracle-database. [Kasutatud 20 11 2021].

[37] K. Sharma, „What is PostgreSQL and why do enterprise developers and start-ups love it?,“ Ubuntu, 19 1 2021. [Võrgumaterjal]. Available: https://ubuntu.com/blog/what-is-postgresql. [Kasutatud 20 11 2021].

[38] K. Campusano, „Building REST APIs with .NET 5, ASP.NET Core, and PostgreSQL,“ End Point Dev, 9 7 2021. [Võrgumaterjal]. Available: https://www.endpointdev.com/blog/2021/07/dotnet-5-web-api/. [Kasutatud 20 11 2021].

[39] M. Gaberov, „How to Build an SPA with Vue.js and C# Using .NET Core,“ freeCodeCamp, 15 9 2020. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/how-to-build-an-spa-with-vuejs-and-c-using-net-core/. [Kasutatud 20 11 2021].

[40] K. Muldma, „Tervisliku toitumise motiivid ja barjäärid,“ 2018. [Võrgumaterjal]. Available: https://digikogu.taltech.ee/et/Item/f2ad4366-4390-4c53-bb9a-aac4ac8945cc. [Kasutatud 6 11 2021].

[41] „Why Jira Software Became So Popular,“ TutorialsPoint. [Võrgumaterjal]. Available: https://www.tutorialspoint.com/why-jira-software-became-so-popular. [Kasutatud 6 11 2021].

[42] „Best C# IDE, that you can use in 2021 - Top Lists of C# IDE,“ Teckangaroo, 25 9 2021. [Võrgumaterjal]. Available: https://teckangaroo.com/best-ide-for-c-that-you-can-use-in-2020/. [Kasutatud 6 11 2021].

[43] A. Mkhitaryan, „Why Is C# Among The Most Popular Programming Languages in The World?“ Medium, 13 10 2017. [Võrgumaterjal]. Available: https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb. [Kasutatud 20 11 2021].

[44] M. Watson, „C# Programming Language“ Stackify, 18 9 2020. [Võrgumaterjal]. Available: https://stackify.com/what-is-c-used-for/. [Kasutatud 20 11 2021].

[45] „PHP The Right Way“ PHP:The Right Way. [Võrgumaterjal]. Available: https://phptherightway.com/. [Kasutatud 20 11 2021].

[46] „16 Top Java Communities, Forums and Groups: The Ultimate Guide“ Lightrun, 29 7 2020. [Võrgumaterjal]. Available: https://lightrun.com/java/16-top-java-communities-forums-and-groups-the-ultimate-guide/. [Kasutatud 20 11 2021].

[47] N. Mukherjee „4 reasons why JavaScript is so popular“ OpenSource, 2 11 2020. [Võrgumaterjal]. Available: https://opensource.com/article/20/11/javascript-popular. [Kasutatud 20 11 2021].

[48] Nikhil „13 Best JavaScript Frameworks For 2020“ LambdaTest, 8 6 2020. [Võrgumaterjal]. Available: https://www.lambdatest.com/blog/best-javascript-framework-2020/. [Kasutatud 20 11 2021].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Raul Vesinurm

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Fitlapi kasutajate toitude tellimise ja müümise veebirakendus“, mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

03.01.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.